4.1 INTRODUCTION

UQ. 4.1.1 *Define Bayes Theorem.* **SPPU - Q. 4(a), May 19, 2 Marks**

UQ. 4.1.2 *Justify with example. The following statement: Naive Bayes are a family of powerful and easy-to-train classifiers that determine the probability of an outcome given a set of conditions using Baye's theorem.* **SPPU - Q. 4(a), Dec. 19, 8 Marks**

- Naive Bayes (NB) algorithm is a supervised classification function which carries out discriminant analysis. It is a generative model which returns probability values as output.
- Naive Bayes algorithm works contrary to the classification strategy of a one Rule classifier. All attributes contribute equally and independently to the decision. The Naive Bayes algorithm makes predictions using Bayes' Theorem, which derives the probability of a prediction from the underlying evidence, as observed in the labelled data.
- Naive Bayes works surprisingly well even if the attribute independence assumption is violated because the classification task does not need accurate probability estimates as long as the greatest probability is associated to the correct class.
- NB affords fast model building and scoring and can be used for both binary and multi-class classification problems. Owing to its light-weight nature and faster building, it is often used in text categorization tasks.
- The Naive Bayes classifier is very useful in high-dimensional problems because multivariate methods like QDA and even LDA might break down in such cases.

Assumptions

The fundamental assumption made by the Naive Bayes algorithm is that each attribute makes an independent and equal contribution to the outcome.

Independent

- All features contribute independently to the given target label Y.
- The Naive Bayes algorithm assumes that the probabilities of the different events (i.e. predictor attributes) are completely independent given the class value. (i.e., knowing the value of one attribute says nothing about the value of another if the class label is known)
- The attributes must be independent otherwise they might have a considerable influence on the accuracy. A relation between input and target variable may lead to biased decisions. The independence assumption is not always correct but it often works well in practice.
- You can model the dependency of an attribute by copying it multiple times. Continuing to make copies of an attribute gives it an increasingly stronger contribution to the decision until the other attributes do not influence at all.

Equal

The event (i.e. predictors attributes) are equally important a priori. If the accuracy remains the same when an attribute is removed, it seems that this attribute is irrelevant to the outcome of the class (target).

UNIT IV

Description

Formula

- The Naive Bayes algorithm is based on conditional probabilities. It uses the Bayes' Theorem, a formula that calculates a probability by counting the frequency of values and combinations of values in the historical data.
- Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred.
- Bayes' theorem: P(A|B)

$$P(A|B) = P(A \cap B) / P(B)$$

Where : A represents the dependent event: Target attribute and B represents the prior event: Predictor attribute.

P(B) is a priori probability of B (The prior probability) wiz. the probability of an event before evidence is seen. The evidence is an attribute value of an unknown instance.

P(A|B) is a posteriori probability of A. It is the probability of an event after evidence is seen. Posteriori = afterwards, after the evidence

Eg: Consider the following data from a high school :

	Female	Male	Total
Teacher	8	12	20
Student	32	48	80
Total	40	60	100

Example 4.1.1

What is the probability that a member of the school is a teacher given the member is Male?

☑ **Solution :**

We want to calculate P(Teacher | Male)

Using the Bayes theorem formula,

$$P(\text{Teacher} \mid \text{Male}) = P(\text{Teacher} \cap \text{Male}) / P(\text{Male})$$

$$P(\text{Teacher} \cap \text{Male}) = 12 / 100 = 0.12$$

$$P(\text{Male}) = 60 / 100 = 0.6$$

Therefore,

$$P(\text{Teacher} \mid \text{Male}) = 0.12 / 0.6 = 0.2$$

Thus the probability of a member being teacher given he is male is 0.2.

Naive

"Naïve" assumption: Evidence splits into independent parts

$$P(B|A) = P(A_1|B)\, P(A_2|B) \ldots\ldots P(A_n|B)\, P(B)$$

Where $A_1, A_2, \ldots, A_n$ are independent priori.

Example

- Spam Filtering with Naïve Bayes on a two classes problem (spam and not spam.)
- The probability that an email is span given the email words is:

P(spam|words) = P(spam).P(words|spam)/P(words)

P(spam|words) ∝ P(spam).P(viagra,rich,…,friend|spam)

Where ∝ is the proportion symbol and (Viagra,rich,...friend) is the list of words.

Using the Chain rule:

One Time

P(spam|words)∝P(spam).P(viagra|spam).P(rich,…,friend|spam,viagra)

Two Time

P(spam|words)∝P(spam).P(viagra|spam).P(rich|spam,viagra).P(…,friend|spam,viagra,rich)

Saying that the word events are completely independent permits us to simplify the above formula to a sequential use of the Bayes' formula such as:

P(spam|words)∝P(viagra|spam).P(rich|spam)...P(friend|spam)

UNIT IV

Discriminant Function

This is the discriminant function for the kth class for naive Bayes. You compute one of these for each of the classes, and then assign the class with the maximum value.

δk(x) ≈ log[πk∏j=1pfkj(xj)]=−12∑j=1p(xj−μkj)2σ2kj+log(πk)

There is

(xj−μkj)2σ2kj - A determinant term which is a contribution of the feature from the mean for the class scaled by the variance log(πk) - A prior term

NB can be used for mixed feature vectors (qualitative and quantitative).

For the quantitative ones, we use the Gaussian. For the qualitative ones, we replace the density function fkj(xj) with probability mass function (histogram) over discrete categories.

Bayes Theorem (Two events)

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A)\,P(B|A)}{P(A)\,P(B|A) + P(A^c)\,P(B|A^c)}$$

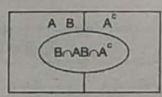

Fig. 4.1.1 : Bayes Theorem

Scanned with CamScanner

We want to know

$$P(A|B) = \frac{P(B \cap A)}{P(B)}$$

Since

$$P(B \cap A) = P(A)\,P(B|A)$$

and

$$P(B) = P(B \cap A) + P(B \cap A^c) = P(A)\,P(B|A) + P(A^c)\,P(B|A^c)$$

thus,

$$P(A|B) = \frac{P(B \cap A)}{P(B)} = \frac{P(B \cap A)}{P(B \cap A) + P(B \cap A^c)} = \frac{P(A)\,P(B|A)}{P(A)\,P(B|A) + P(A^c)\,P(B|A^c)}$$

Bayes's Theorem (general)

Let $A_1, A_2, \ldots, A_n$ be mutually exclusive events and

$$A_1 \cup A_2 \cup \ldots A_n = S$$

then

$$P(A_i|B) = \frac{P(A_i \cap B)}{P(B)} = \frac{P(A_i)\,P(B|A_i)}{P(A_1)\,P(B|A_1) + P(A_2)\,P(B|A_2) + \ldots + P(A_n)\,P(B|A_n)}$$

$$i = 1, 2, \ldots, n$$

4.2 PRINCIPLE OF NAIVE BAYES CLASSIFIER

UQ. 4.2.1 *Elaborate Naive Bayes Classifier working with example.* **SPPU - Q. 4(a). May 19, 6 Marks**

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification tasks. The crux of the classifier is based on the Bayes theorem.

Bayes Theorem

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

- Using Bayes theorem, we can find the probability of **A** happening, given that **B** has occurred. Here, **B** is the evidence and **A** is the hypothesis. The assumption made here is that the predictors/features are independent. That is the presence of one particular feature does not affect the other. Hence it is called naive.

Example

Let us take an example to get some better intuition. Consider the problem of playing golf. The dataset is represented as below :

	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

- We classify whether the day is suitable for playing golf, given the features of the day. The columns represent these features and the rows represent individual entries.
- If we take the first row of the dataset, we can observe that is not suitable for playing golf if the outlook is rainy, the temperature is hot, humidity is high and it is not windy.
- We make two assumptions here, one as stated above we consider that these predictors are independent. That is, if the temperature is hot, it does not necessarily mean that the humidity is high.
- Another assumption made here is that all the predictors have an equal effect on the outcome. That is, the day being windy does not have more importance in deciding to play golf or not.
- According to this example, Bayes theorem can be rewritten as :

$$P(y|X) = \frac{P(X|y)\,P(y)}{P(X)}$$

- The variable y is the class variable (play golf), which represents if it is suitable to play golf or not given the conditions. Variable **X** represent the parameters/features.

X is given as,

$$X = (x_1, x_2, x_3, \ldots, x_n)$$

- Here $x_1, x_2, \ldots, x_n$ represent the features, i.e. they can be mapped to outlook, temperature, humidity and windy. By substituting for X and expanding using the chain rule we get,

$$P(y|x_1, \ldots, x_n) = \frac{P(x_1|y)\,P(x_2|y)\ldots P(x_n|y)\,P(y)}{P(x_1)\,p(x_2)\ldots P(x_n)}$$

UNIT IV

- Now, you can obtain the values for each by looking at the dataset and substitute them into the equation. For all entries in the dataset, the denominator does not change, it remains static. Therefore, the denominator can be removed and proportionality can be introduced.

$$P(y|x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

- In our case, the class variable (y) has only two outcomes, yes or no. There could be cases where the classification could be multivariate. Therefore, we need to find the class y with maximum probability.

$$y = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

- Using the above function, we can obtain the class, given the predictors.

Example 4.2.1

Say you have 1000 fruits which could be either 'banana', 'orange' or 'other'. These are the 3 possible classes of the Y variable. The data distribution is as shown in the table. The objective of the classifier is to predict if a given fruit is a 'Banana' or 'Orange' or 'Other' given that it is Long, Sweet and Yellow.

☑ **Solution :**

The idea is to compute the 3 probabilities, that is the probability of the fruit being a banana, orange or other. Whichever fruit type gets the highest probability wins.

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	Total
Banana	400	100	350	150	450	50	500
Orange	0	3000	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

▶ **Step 1 : Compute the 'Prior' probabilities for each of the class of fruits.**

That is, the proportion of each fruit class out of all the fruits from the population. You can provide the 'Priors' from prior information about the population. Otherwise, it can be computed from the training data.

For this case, let's compute from the training data. Out of 1000 records in training data, you have 500 Bananas, 300 Oranges and 200 Others.

P(Y = Banana) = 500 / 1000 = 0.50

P(Y = Orange) = 300 / 1000 = 0.30

P(Y = Other) = 200 / 1000 = 0.20

▶ **Step 2 : Compute the probability of evidence that goes in the denominator.**

This is nothing but the product of P of Xs for all X. This is an optional step because the denominator is the same for all the classes and so will not affect the probabilities.

P(x1 = Long) = 500 / 1000 = 0.50

P(x2 = Sweet) = 650 / 1000 = 0.65

P(x3 = Yellow) = 800 / 1000 = 0.80

▶ **Step 3 : Compute the probability of likelihood of evidences that goes in the numerator.**

It is the product of conditional probabilities of the 3 features. If you refer back to the formula, it says P(X1 |Y). Here X1 is 'Long' and say Y is 'Banana'. That means the probability the fruit is 'Long' given that it is a Banana. In the above table, you have 500 Bananas. Out of that 400 is long. So, P(Long | Banana) = 400/500 = 0.8.

Here, we have only considered Banan.

Probability of Likelihood for Banana

P(x1 = Long | Y = Banana) = 400 / 500 = 0.80

P(x2 = Sweet | Y = Banana) = 350 / 500 = 0.70

P(x3 = Yellow | Y = Banana) = 450 / 500 = 0.90

Finally, substituting the three values in Naïve Bayes equation,

P(Banana | Long, sweet and yellow) = (P(Long | Banana) * P(Sweet | Banana) * P(Yellow | Banana) * P(Banana)) /(P(Long) * P(Sweet) * P(Yellow))

= 0.8*0.7*0.9*0.5/P(Evidence)

= 0.252/P(Evidence)

Where P(Evidence) = P(Long) * P(Sweet) * P(Yellow) (Constant denominator value)

Similarly, P(Orange | Long, Sweet and yellow) = 0 (because P(Long | Orange) = 0)

And P(Other fruit | Long, Sweet and yellow) = 0.01875 / P(Evidence)

Therefore, the given fruit which is long, sweet and yellow is Banana as it has the highest probability value amongst all the given classes

UNIT IV

4.3 TYPES OF NAIVE BAYES CLASSIFIER

UQ. 4.3.1 *Write shorts notes on:*

i) Bernoulli naive Bayes.

ii) multinomial naive Bayes.

iii) Gaussian naive Bayes.

SPPU - Q. 3(c), May 19, Q. 3(b), Dec. 19, 9 Marks

1. Multinomial Naive Bayes
2. Bernoulli Naive Bayes
3. Gaussian Naive Bayes

1. Multinomial Naive Bayes

This is mostly used for document classification problem, i.e. whether a document belongs to the category of sports, politics, technology, etc. The features/predictors used by the classifier are the frequency of the words present in the document.

2. Bernoulli Naive Bayes

This is similar to the multinomial naive bayes but the predictors are boolean variables. The parameters that we use to predict the class variable take up only values yes or no, for example, if a word occurs in the text or not.

3. Gaussian Naive Bayes

When the predictors take up a continuous value and are not discrete, we assume that these values are sampled from a gaussian distribution.

Since the way the values are present in the dataset changes, the formula for conditional probability changes to,

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

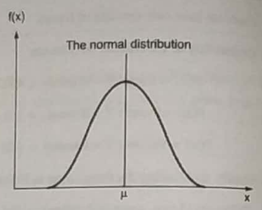

Fig. 4.3.1 : Gaussian Distribution (Normal Distribution)

4.4 SCIKIT LEARN PERSPECTIVE OF NAÏVE BAYES

UQ. 4.4.1 Write shorts notes on:
i) Bernoulli naive Bayes.
ii) multinomial naive Bayes.
iii) Gaussian naive Bayes.
SPPU - Q. 3(c), May 19, Q. 3(b), Dec. 19 9 Marks

- Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one-dimensional distribution. This, in turn, helps to alleviate problems stemming from the curse of dimensionality.
- On the flip side, although naive Bayes is known as a decent classifier, it is known to be a bad estimator, so the probability outputs from predict_proba are not to be taken too seriously.

4.4.1 Gaussian Naive Bayes

Gaussian NB implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian :

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

The parameters σ_y and μ_y are estimated using maximum likelihood.

```
>>>from sklearn import datasets
>>>iris=datasets.load_iris()
>>>from sklearn.naive_bayes import GaussianNB
>>>gnb=GaussianNB()
>>>y_pred=gnb.fit(iris.data,iris.target).predict(iris.data)
>>>print("Number of mislabeled points out of a total %d points : %d"
... %(iris.data.shape[0],(iris.target!=y_pred).sum()))
Number of mislabeled points out of a total 150 points : 6
```

4.4.2 Multinomial Naive Bayes

- Multinomial NB implements the naive Bayes algorithm for multinomially distributed data and is one of the two classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts, although tf-idf vectors are also known to work well in practice). The distribution is parameterized by vectors $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$ for each class y, where n is the number of features (in text classification, the size of the vocabulary) and θ_{yi} is the probability $P(x_i|y)$ of feature i appearing in a sample belonging to class y.
- The parameters θ_y is estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting :

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

Where $N_{yi} = \Sigma\, x \in T\, x_i$ is the number of times feature i appears in a sample of class y in the training set T, and $N_y = \Sigma\, n_i = 1\, N_{yi}$ is the total count of all features for class y.

- The smoothing priors $\alpha \geq 0$ accounts for features not present in the learning samples and prevents zero probabilities in further computations. Setting $\alpha = 1$ is called Laplace smoothing, while $\alpha < 1$is called Lidstone smoothing.

Example of Multinomial NB in scikit learn

```
>>> import numpy as np
>>> rng = np.random.RandomState(1)
>>> X = rng.randint(5, size=(6, 100))
>>> y = np.array([1, 2, 3, 4, 5, 6])
>>> from sklearn.naive_bayes import MultinomialNB
>>> clf = MultinomialNB()
>>> clf.fit(X, y)
MultinomialNB()
>>> print(clf.predict(X[2:3]))
[3]
```

UNIT IV

4.4.3 Complement Naive Bayes

- ComplementNB implements the complement naive Bayes (CNB) algorithm. CNB is an adaptation of the standard multinomial naive Bayes (MNB) algorithm that is particularly suited for imbalanced data sets.
- Specifically, CNB uses statistics from the complement of each class to compute the model's weights. The inventors of CNB show empirically that the parameter estimates for CNB are more stable than those for MNB.
- Further, CNB regularly outperforms MNB (often by a considerable margin) on text classification tasks. The procedure for calculating the weights is as follows :

$$\hat{\theta}_{ci} = \frac{\alpha_i + \sum_{j:y_j \neq c} d_{ij}}{\alpha + \sum_{j:y_j \neq c} \sum_k d_{kj}}$$

$$w_{ci} = \log \hat{\theta}_{ci}$$

$$w_{ci} = \frac{w_{ci}}{\sum_j | w_{cj} |}$$

where the summations are over all documents j not in class c, d_{ij} is either the count or tf-idf value of term i in document j, α_i is a smoothing hyper parameter like that found in MNB, and $\alpha = \sum i \, \alpha_i$. The second normalization addresses the tendency for longer documents to dominate parameter estimates in MNB. The classification rule is :

$$\hat{c} = \arg\min_c \sum_i t_i w_{ci}$$

i.e., a document is assigned to the class that is the poorest complement match.

4.4.4 Bernoulli Naive Bayes

- BernoulliNB implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions; i.e., there may be multiple features but each one is assumed to be a binary-valued (Bernoulli, boolean) variable. Therefore, this class requires samples to be represented as binary-valued feature vectors; if handed any other kind of data, a BernoulliNB instance may binarize its input (depending on the binarize parameter).
- The decision rule for Bernoulli naive Bayes is based on

$$P(x_i \mid y) = P(i \mid y) x_i + (1 - P(i \mid y)) (1 - x_i)$$

- Which differs from multinomial NB's rule in that it explicitly penalizes the non-occurrence of a feature i that is an indicator for class y, where the multinomial variant would simply ignore a non-occurring feature.
- In the case of text classification, word occurrence vectors (rather than word count vectors) may be used to train and use this classifier. BernoulliNB might perform better on some datasets, especially those with shorter documents. It is advisable to evaluate both models if time permits.

Out-of-core naive Bayes model fitting

- Naive Bayes models can be used to tackle large scale classification problems for which the full training set might not fit in memory. To handle this case, MultinomialNB, BernoulliNB, and GaussianNB expose a partial_fit method that can be used incrementally as done with other classifiers. All naive Bayes classifiers support sample weighting.

- Contrary to the fit method, the first call to partial_fit needs to be passed the list of all the expected class labels.
- The partial_fit method call of naive Bayes models introduces some computational overhead. It is recommended to use data chunk sizes that are as large as possible, that is as the available RAM allows.

4.5 SUPPORT VECTOR MACHINE

UQ. 4.5.1 *What are Linear support vector machines? Explain with example.* SPPU - Q. 4(b), May 19, 4 Marks

UQ. 4.5.2 *What do you mean by Support Vector machine? Explain with example.* SPPU - Q. 3(a), Dec. 19, 8 Marks

- A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane.
- In other words, given a labelled training dataset (as used in supervised learning), the algorithm outputs an optimal hyperplane which categorizes any new test examples. In 2D space, this hyperplane is a line dividing the plane into two parts where the two output classes lie on either side.
- Suppose you are given a plot of two label classes on the graph as shown in Fig. 4.5.1. Can you decide a separating line for the classes?

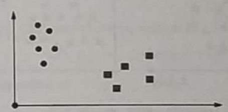

Fig. 4.5.1: Draw a line that separates circles and squares

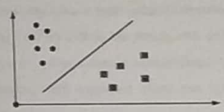

Fig. 4.5.2 : Possible line which can be drawn

UNIT IV

- You might come up with something similar to the following (Fig. 4.5.2). It fairly separates the two classes. Any point that is on the left side of the line falls into the circle class, and the ones on the right fall into the square class. **Separation of classes :** That is what SVM does. It finds out a line/ hyper-plane (in multidimensional space) that separate outs classes.

4.5.1 How does SVM Work?

The main objective is to segregate the given dataset in the best possible way. The distance between the either nearest points is known as the margin. The objective is to select a hyperplane with the maximum possible margin between support vectors in the given dataset. SVM searches for the maximum marginal hyperplane in the following steps :

1. Generate hyperplanes which segregate the classes in the best way. Left-hand side Fig. 4.5.3 showing three hyperplanes. Here, two have higher classification error but one hyperlpane separating two classes correctly.
2. Select the right hyperplane with the maximum segregation from either nearest data points as shown in the right-hand side Fig. 4.5.3.

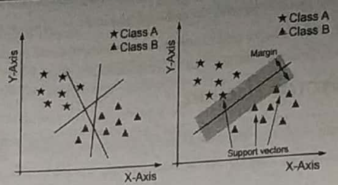

Fig. 4.5.3 : Support Vector Machine

4.5.2 Dealing with Non-Linear and Inseparable Planes

UQ. 4.5.3 *Explain the non-linear SVM with example.* **SPPU - Q. 3(b), May 19, 5 Marks**

Some problems can't be solved using linear hyperplane, as shown in the Fig. 4.5.4 (left-hand side). In such a situation, SVM uses a kernel trick to transform the input space to a higher dimensional space as shown on the right. The data points are plotted on the x-axis and z-axis (Z is the squared sum of both x and y : z = x ^ 2 = y ^ 2). Now you can easily segregate these points using linear separation.

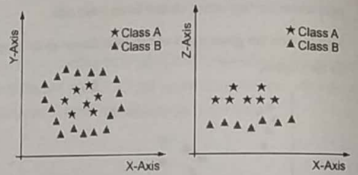

Fig. 4.5.4 : Linear vs. Non linear sample

4.5.3 SVM Kernels

- The SVM algorithm is implemented in practice using a kernel. A kernel transforms an input data space into the required form. SVM uses a technique called the kernel trick.
- Here, the kernel takes a low-dimensional input space and transforms it into a higher-dimensional space. In other words, you can say that it converts non-separable problems to separable problems by adding more dimensions to it.
- It is most useful in non-linear separation problems. Kernel trick helps to build a more accurate classifier.
- **Linear Kernel :** A linear kernel can be used as a normal dot product of any two given observations. The product between two vectors is the sum of the multiplication of each pair of input values.

$$K(x, x_i) = sum(x * x_i)$$

- **Polynomial Kernel :** A polynomial kernel is a more generalized form of the linear kernel. The polynomial kernel can distinguish curved or nonlinear input space.

$$K(x, x_i) = 1 + sum(x * x_i) \wedge d$$

Where d is the degree of the polynomial. d=1 is similar to the linear transformation. The degree needs to be manually specified in the learning algorithm.

- **Radial Basis Function Kernel :** The Radial basis function kernel is a popular kernel function commonly used in support vector machine classification. RBF can map an input space in infinite-dimensional space.

$$K(x, x_i) = exp(- gamma * sum((x - x_i)^2))$$

- Here gamma is a parameter, which ranges from 0 to 1. A higher value of gamma will perfectly fit the training dataset, which causes over-fitting. Gamma = 0.1 is considered to be a good default value. The value of gamma needs to be manually specified in the learning algorithm.
- **Tuning parameters :** Kernel, Regularization, Gamma and Margin.

4.5.4 Kernel

- The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra. This is where the kernel plays a role.
- For **linear kernel** the equation for prediction for a new input using the dot product between the input (x) and each support vector (x_i) is calculated as follows:

$$f(x) = B(0) + sum(a_i * (x, x_i))$$

- This is an equation that involves calculating the inner products of a new input vector (x) with all support vectors in training data. The coefficients B_0 and a_i (for each input) must be estimated from the training data by the learning algorithm.

The polynomial kernel can be written as

$$K(x, x_i) = 1 + sum(x * x_i) \wedge d$$

and **exponential** as

$$K(x, x_i) = exp(- gamma * sum(x - x_i)^2)$$

- Polynomial and exponential kernels calculate the separation line in a higher dimension. This is called **kernel trick**.

UNIT IV

4.5.5 Regularization

- The Regularization parameter (often termed as C parameter in python's sklearn library) tells the SVM optimization how much you want to avoid misclassifying each training example.
- For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.
- The Fig. 4.5.5 and Fig. 4.5.6 are examples of two different regularization parameters. Left one has some misclassification due to lower regularization value. Higher value leads to results like the right one.

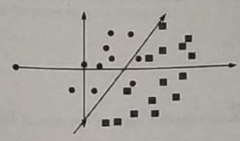

Fig. 4.5.5 : Low regularization value,

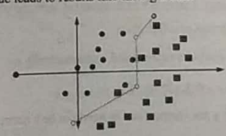

Fig. 4.5.6 : High regularization value

4.5.6 Gamma

- The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'.
- In other words, with low gamma, points far away from plausible separation line are considered in calculation for the separation line. Whereas high gamma means the points close to plausible line are considered in calculation.

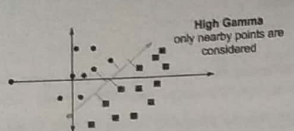

Fig. 4.5.7 : . High Gamma

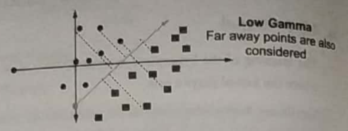

Fig. 4.5.8 : Low Gamma

4.5.7 Margin

- And finally last but very important characteristic of SVM classifier. SVM to core tries to achieve a good margin.
- A margin is a separation of line to the closest class points.
- A good margin is one where this separation is larger for both the classes. Fig. 4.5.9 and Fig. 4.5.10 below depict visually a good and bad margin. A good margin allows the points to be in their respective classes without crossing to the other side.

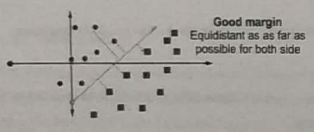

Fig. 4.5.9 : Good margin

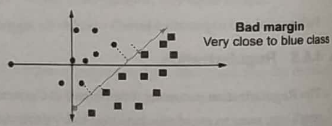

Fig. 4.5.10 : Bad Margin

4.5.8 Linear Support Vector Machine

- Linear Support Vector Machines involve finding a hyperplane to distinguish the two classes. Given a dataset consisting of 'n' samples, each of these will have their output variable y to be equal to 1 or -1, indicating the class to which the sample belongs. Linear SVM is modeled with the objective of finding the hyperplane with the maximum margin from the nearest points of either class.
- Hyperplane could be represented mathematically as:

$$w.x - b = 0$$

where w and x are vectors, and w need not be a normal vector always.

- If the training data is linearly separable, two hyperplanes that separate the two classes of data can be selected such that the distance between them is maximum. The maximum margin hyperplane would then lie midway between these two planes.
- In case the data is not linearly separable, the hinge loss function could be used in the optimization objective.

4.5.9 Non-Linear SVM

UQ. 4.5.4	What problems are faced by SVM when used with real datasets?	SPPU - Q. 3(a), May 19, 3 Marks
UQ. 4.5.5	Explain the non-linear SVM with example.	SPPU - Q. 3(b), May 19, 5 Marks

- SVM is a linear classifier that learns an (n – 1)-dimensional classifier for classification of data into two classes. However, it can be used for classifying a non-linear dataset. This can be done by projecting the dataset into a higher dimension in which it is linearly separable.
- A trick known as "kernel trick" is used to learn a linear classifier to classify a non-linear dataset. It transforms the linearly inseparable data into a linearly separable one by projecting it into a higher dimension.
- A kernel function is applied to each data instance to map the original non-linear data points into some higher dimensional space in which they become linearly separable.

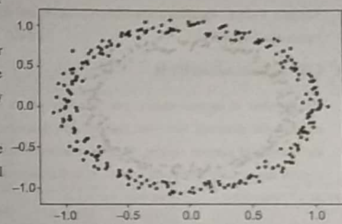

Fig. 4.5.11 : 2D Non Linear SVM

Example

- Consider the given data which is clearly non-linear dataset and consists of two features (say, X and Y). In order to use SVM for classifying this data, introduce another feature $Z = X^2 + Y^2$ into the dataset.
- Thus, projecting the 2-dimensional data into 3-dimensional space. The first dimension representing the feature X, second representing Y and third representing Z (which, mathematically, is equal to the radius of the circle of which the point (x, y) is a part of).

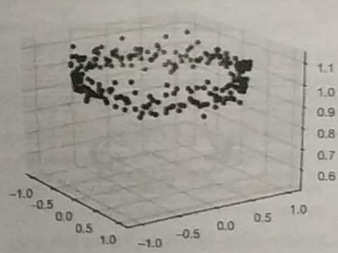

Fig. 4.5.12 : 3D Non Linear SVM

- Now it can be seen that the data is clearly separable by a 2D hyperplane.

UNIT IV

Problems when using SVM on real-world data

- SVMs are less effective if there is noisy data which is the case in most real-world tasks. Extra preprocessing of data is required.
- Real-world data often contains overlapping points. As a result, there can be issues in drawing a clear hyperplane without misclassification.
- SVM is not very robust to outliers in its basic form. The presence of a few outliers can lead to very bad global misclassification results.
- It is difficult to understand and interpret the variable weights and individual impact especially in case of large datasets.
- Deciding the ideal kernel function for the data in hand can be a challenging task.
- Small calibrations cannot be done easily to the model and thus it is difficult to incorporate any business logic which the organization might intend to add.
- Support Vector Machines are memory intensive and don't scale well to larger datasets. Random forests turn out to be a better alternative

4.5.10 Controlled SVM

- The number of support vectors can be controlled. In real-world problems, we can extract a lot of support vectors to increase the accuracy. However, this can slow down the whole process. NuSVC allows a tradeoff between precision and the number of support vectors. NuSVC is an implementation available in scikit-learn. The parameter nu can be used to control the number of support vectors as well as the training errors.
- Nu has a value between (0,1]
  - o A greater Nu value will increase the number of support vectors.
  - o A smaller Nu value reduces the fraction of errors by the model.
- In short, nu is an upper bound on the fraction of training errors and a lower bound of the fraction of support vectors.

4.5.11 Kernel Based Classification

UQ. 4.5.6 *Write a short note on : Kernel-based classification.* **SPPU - Q. 4(b)(ii), Dec. 19, 5 Marks**

- SVM algorithms use a set of mathematical functions that are defined as the kernel. The function of kernel is to take data as input and transform it into the required form. Different SVM algorithms use different types of kernel functions.
- These functions can be different types. For example linear, nonlinear, polynomial, radial basis function (RBF), and sigmoid.
- Introduce Kernel functions for sequence data, graphs, text, images, as well as vectors. The most used type of kernel function is RBF. Because it has localized and finite response along the entire x-axis.

1. **Polynomial kernel**

$$L(x_i, x_j) = (x_i \cdot x_j + 1)^d$$

Where d is the degree of polynomial. Quite popular in image processing

2. **Gaussian kernel**

It is a general-purpose kernel; used when there is no prior knowledge about the data.

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$$

It is parameterized for the inner gamma > 0 or sometimes for gamma = $1/2\sigma^2$

3. **Laplace RBF kernel**

$$k(x, y) = \exp\left(\frac{-\|x - y\|}{\sigma}\right)$$

4. **Hyperbolic tangent kernel**

$$k(x_i, x_j) = \tan h\,(kX_i \cdot X_j + c)$$

It can be used in neural networks for some(not every) k>0 and c<0

5. **Sigmoid kernel**

We can use it as a proxy for neural networks. Equation is

$$k(x, y) = \tanh(\alpha\, x^T y + c)$$

4.6 SCIKIT LEARN PERSPECTIVE OF SVM

GQ. 4.6.1 *How to implement SVM using python sklearn library ?*

- The support vector machines in scikit-learn support both dense and sparse sample vectors as input. However, to use an SVM to make predictions for sparse data, it must have been fit on such data. For optimal performance, use C-ordered numpy.ndarray (dense) or scipy.sparse.csr_matrix (sparse) with dtype = float64.
- SVC, NuSVC and LinearSVC are classes capable of performing multi-class classification on a dataset. SVC and NuSVC are similar methods, but accept slightly different sets of parameters and have different mathematical formulations. On the other hand, Linear SVC is another implementation of Support Vector Classification for the case of a linear kernel. Note that Linear SVC does not accept keyword kernel, as this is assumed to be linear. It also lacks some of the members of SVC and NuSVC, like support_.

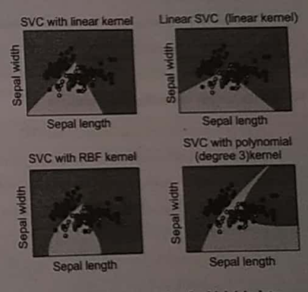

Fig. 4.6.1 : SVM Representation for high iris data

UNIT IV

- As other classifiers, SVC, NuSVC, and LinearSVC take as input two arrays: an array X of size [n_samples, n_features] holding the training samples, and an array y of class labels (strings or integers), size [n_samples] :

```
>>>from sklearn import svm
>>>X=[[0,0],[1,1]]
>>>y=[0,1]
```

```
>>>clf=svm.SVC(gamma='scale')
>>>clf.fit(X,y)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
max_iter=-1, probability=False, random_state=None, shrinking=True,
tol=0.001, verbose=False)
```

- After being fitted, the model can then be used to predict new values :

```
>>>clf.predict([[2.,2.]])
array([1])
```

- SVMs decision function depends on some subset of the training data, called the support vectors. Some properties of these support vectors can be found in members support_vectors_, support_ and n_support:

```
>>># get support vectors
>>>clf.support_vectors_
array([[0., 0.],
[1., 1.]])
>>># get indices of support vectors
>>>clf.support_
array([0, 1]...)
>>># get number of support vectors for each class
>>>clf.n_support_
array([1, 1]...)
```

SVMs Linear classification example

```
>>> from sklearn.svm import LinearSVC
>>> from sklearn.datasets import make_classification
>>> X, y = make_classification(n_features=4, random_state=0)
>>> clf = LinearSVC(random_state=0, tol=1e-5)
>>> clf.fit(X, y)
LinearSVC(random_state=0, tol=1e-05)
>>> print(clf.coef_)
[[0.085... 0.394... 0.498... 0.375...]]
>>> print(clf.intercept_)
[0.284...]
>>> print(clf.predict([[0, 0, 0, 0]]))
[1]
```


4.7 KERNEL FUNCTIONS

GQ. 4.7.1 *Describe importance of kernel function in classification problems.*

- The kernel function can be any of the following :
  - linear : $\langle x, x' \rangle$.
  - polynomial : $(\gamma \langle x, x' \rangle + r)$ d. d is specified by keyword degree, r by coef0.
  - rbf : exp(− γ||x − x′ || 2). Γ is specified by keyword gamma, must be greater than 0.
  - sigmoid (tanh(γ ⟨x, x′⟩ + r)), where r is specified by coef0.
- Different kernels are specified by keyword kernel at initialization :

```
>>>linear_svc=svm.SVC(kernel='linear')
>>>linear_svc.kernel
'linear'
>>>rbf_svc=svm.SVC(kernel='rbf')
>>>rbf_svc.kernel
'rbf'
```

Custom Kernels

- You can define your own kernels by either giving the kernel as a python function or by precomputing the Gram matrix.
- Classifiers with custom kernels behave the same way as any other classifiers, except that :
  - Field support_vectors_ is now empty, only indices of support vectors are stored in support_
  - A reference (and not a copy) of the first argument in the fit() method is stored for future reference. If that array changes between the use of fit() and predict() you will have unexpected results.

UNIT IV

Using Python functions as kernels

- You can also use your own defined kernels by passing a function to the keyword kernel in the constructor.
- Your kernel must take as arguments two matrices of shape (n_samples_1, n_features), (n_samples_2, n_features) and return a kernel matrix of shape (n_samples_1, n_samples_2).
- The following code defines a linear kernel and creates a classifier instance that will use that kernel:

```
>>>import numpy as np
>>>from sklearn import svm
>>>def my_kernel(X,Y):
... returnnp.dot(X,Y.T)
...
>>>clf=svm.SVC(kernel=my_kernel).
```

Using the Gram matrix

Set kernel='precomputed' and pass the Gram matrix instead of X in the fit method. At the moment, the kernel values between all training vectors and the test vectors must be provided.

```
>>>import numpy as np
>>>from sklearn import svm
>>>X=np.array([[0,0],[1,1]])
>>>y=[0,1]
>>>clf=svm.SVC(kernel='precomputed')
>>># linear kernel computation
>>>gram=np.dot(X,X.T)
>>>clf.fit(gram,y)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
kernel='precomputed', max_iter=-1, probability=False,
random_state=None, shrinking=True, tol=0.001, verbose=False)
>>># predict on training examples
>>>clf.predict(gram)
array([0, 1])
```

Parameters of the RBF Kernel

When training an SVM with the Radial Basis Function (RBF) kernel, two parameters must be considered : C and gamma. Proper choice of C and gamma is critical to the SVM's performance as discussed earlier. One is advised to use sklearn.model_selection.GridSearchCV with C and gamma spaced exponentially far apart to choose good values.

4.8 MULTI-CLASS CLASSIFICATION USING SVM

GQ. 4.8.1 *How SVM handle multiclass classification problems.*

- SVC and NuSVC implement "one against one" approach for multi-class classification. If n_class is the number of classes, then n_class * (n_class - 1)/2 classifiers are constructed and each one trains data from two classes. To provide a consistent interface with other classifiers, the decision_function_shape option allows to monotically transform the results of the "one-against-one" classifiers to a decision function of shape (n_samples, n_classes).

Example :

```
>>> X = [[0], [1], [2], [3]]
>>> Y = [0, 1, 2, 3]
>>> clf = svm.SVC(decision_function_shape='ovo')
>>> clf.fit(X, Y)
SVC(decision_function_shape='ovo')
```



```
>>> dec = clf.decision_function([[1]])
>>> dec.shape[1] # 4 classes: 4*3/2 = 6
6
>>> clf.decision_function_shape = "ovr"
>>> dec = clf.decision_function([[1]])
>>> dec.shape[1] # 4 classes
4
```

- On the other hand, LinearSVC implements "one-vs-the-rest" multi-class strategy, thus training n_class models. If there are only two classes, only one model is trained :

```
>>> lin_clf = svm.LinearSVC()
>>> lin_clf.fit(X, Y)
LinearSVC()
>>> dec = lin_clf.decision_function([[1]])
>>> dec.shape[1]
4
```

- Note that the LinearSVC also implements an alternative multi-class strategy, the so-called multi-class SVM formulated by Crammer and Singer, by using the option multi_class='crammer_singer'.
- This method is consistent, which is not true for one-vs-rest classification. In practice, one-vs-rest classification is usually preferred, since the results are mostly similar, but the runtime is significantly less.

4.9 SUPPORT VECTOR REGRESSION

UQ. 4.9.1 *Explain with example the variant of SVM, the Support vector regression.* SPPU - Q. 8(c), May 19, 5 Marks

GQ. 4.9.2 *Write short note on : Support vector regression.*

- The method of Support Vector Classification can be extended to solve regression problems. This method is called Support Vector Regression.
- The model produced by support vector classification (as described above) depends only on a subset of the training data because the cost function for building the model does not care about training points that lie beyond the margin.
- Analogously, the model produced by Support Vector Regression depends only on a subset of the training data, because the cost function for building the model ignores any training data close to the model prediction.
- There are three different implementations of Support Vector Regression : SVR, NuSVR, and LinearSVR. LinearSVR provides a faster implementation than SVR but only considers linear kernels, while NuSVR implements a slightly different formulation.

UNIT IV

- As with classification classes, the fit method will take as argument vectors X, y, only that in this case y is expected to have floating point values instead of integer values :

```
>>>from sklearn import svm
>>>X=[[0,0],[2,2]]
>>>y=[0.5,2.5]
>>>clf=svm.SVR()
>>>clf.fit(X,y)
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
gamma='auto_deprecated', kernel='rbf', max_iter=-1, shrinking=True,
tol=0.001, verbose=False)
>>>clf.predict([[1,1]])
array([1.5])
```

☞ **Parameters of SVR as follows :**

1. **Kernel :** Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used.
2. **Degree :** Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.
3. **Gamma :** Kernel coefficient for 'rbf', 'poly' and 'sigmoid'
4. **C :** Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.
5. **Tol :** Tolerance for stopping criterion.
6. **Epsilon :** Epsilon in the epsilon-SVR model. It specifies the epsilon-tube within which no penalty is associated in the training loss function with points predicted within a distance epsilon from the actual value.
7. **Cache_size :** Specify the size of the kernel cache (in MB).
8. **Verbose :** Enable verbose output. Note that this setting takes advantage of a per-process runtime setting in libsvm that, if enabled, may not work properly in a multithreaded context.
9. **Max_iter :** Hard limit on iterations within solver, or -1 for no limit.

☞ **Example of LinearSVR in Scikit Learn**

```
>>> from sklearn.svm import LinearSVR
>>> from sklearn.datasets import make_regression
>>> X, y = make_regression(n_features=4, random_state=0)
>>> regr = LinearSVR(random_state=0, tol=1e-5)
>>> regr.fit(X, y)
```



```
LinearSVR(random_state=0, tol=1e-05)
>>> print(regr.coef_)
[16.35... 26.91... 42.30... 60.47...]
>>> print(regr.intercept_)
[-4.29...]
>>> print(regr.predict([[0, 0, 0, 0]]))
[-4.29...]
```

4.10 UNIVERSITY QUESTIONS AND ANSWERS

May 2019

Q. 4.1 What problems are faced by SVM when used with real datasets? *(Ans. : Refer sections 4.5.9)* **(Q. 3(a), 3 Marks)**

Q. 4.2 Explain the non-linear SVM with example. *(Ans. : Refer sections 4.5.2 and 4.5.9)* **(Q. 3(b), 9 Marks)**

Q. 4.3 Write shorts notes on :
i) Bernoulli naive Bayes.
ii) multinomial naive Bayes.
iii) Gaussian naive Bayes. *(Ans. : Refer sections 4.3 and 4.4)* **(Q. 3(c), 9 Marks)**

Q. 4.4 Define Bayes Theorem. Elaborate Naive Bayes Classifier working with example. *(Ans. : Refer sections 4.1 and 4.2)* **(Q. 4(a), 8 Marks)**

Q. 4.5 What are Linear support vector machines? Explain with example. *(Ans. : Refer section 4.5)* **(Q. 4(b), 4 Marks)**

Q. 4.6 Explain with example the variant of SVM, the Support vector regression. *(Ans. : Refer section 4.9)* **(Q. 4(c), 5 Marks)**

UNIT IV

Dec. 2019

Q. 4.7 What do you mean by Support Vector machine? Explain with example. *(Ans. : Refer section 4.5)* **(Q. 3(a), 8 Marks)**

Q. 4.8 Write shorts notes on :
i) Bernoulli naive Bayes.
ii) multinomial naive Bayes.
iii) Gaussian naive Bayes. *(Ans. : Refer sections 4.4 and 4.4)* **(Q. 3(b), 9 Marks)**

Q. 4.9 Justify with example. The following statement: Naive Bayes are a family of powerful and easy-to-train classifiers that determine the probability of an outcome given a set of conditions using Baye's theorem. *(Ans. : Refer section 4.1)* **(Q. 4(a), 8 Marks)**

Q. 4.10 Write a short note on : Kernel-based classification. *(Ans. : Refer section 4.5.11)* **(Q. 4(b), 5 Marks)**

Chapter Ends...

❑❑❑